####TODO:
https://www.imooc.com/code/563
https://www.imooc.com/code/564
PHP进阶篇-函数： http://www.imooc.com/learn/737
PHP进阶篇-日期时间函数: http://www.imooc.com/learn/698
PHP进阶篇-GD库图像处理: http://www.imooc.com/learn/701
PHP进阶篇-字符串操作: http://www.imooc.com/learn/726

####php相关数据库
1.PHP数据库扩展
PHP中一个数据库可能有一个或者多个扩展，其中既有官方的，也有第三方提供的。像Mysql常用的扩展有原生的mysql库，也可以使用增强版的mysqli扩展，还可以使用PDO进行连接与操作。
不同的扩展提供基本相近的操作方法，不同的是可能具备一些新特性，以及操作性能可能会有所不同。
mysql扩展进行数据库连接的方法：
$link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password');
mysqli扩展：
$link = mysqli_connect('mysql_host', 'mysql_user', 'mysql_password');
PDO扩展
$dsn = 'mysql:dbname=testdb;host=127.0.0.1';
$user = 'dbuser';
$password = 'dbpass';
$dbh = new PDO($dsn, $user, $password);
2.当数据库操作完成以后，可以使用mysql_close关闭数据库连接，默认的，当PHP执行完毕以后，会自动的关闭数据库连接。
虽然PHP会自动关闭数据库连接，一般情况下已经满足需求，但是在对性能要求比较高的情况下，可以在进行完数据库操作之后尽快关闭数据库连接，以节省资源，提高性能。

####cookie
1.Cookie是存储在客户端浏览器中的数据，我们通过Cookie来跟踪与存储用户数据。一般情况下，Cookie通过HTTP headers从服务端返回到客户端。多数web程序都支持Cookie的操作，因为Cookie是存在于HTTP的标头之中，所以必须在其他信息输出以前进行设置，类似于header函数的使用限制。
2.PHP通过setcookie函数进行Cookie的设置，任何从浏览器发回的Cookie，PHP都会自动的将他存储在$_COOKIE的全局变量之中，因此我们可以通过$_COOKIE['key']的形式来读取某个Cookie值。
3.php中没有删除Cookie的函数，在PHP中删除cookie也是采用setcookie函数来实现。setcookie('test', '', time()-1);
4.cookie中的路径用来控制设置的cookie在哪个路径下有效，默认为'/'，在所有路径下都有，当设定了其他路径之后，则只在设定的路径以及子路径下有效。
5.cookie将数据存储在客户端，建立起用户与服务器之间的联系，通常可以解决很多问题，但是cookie仍然具有一些局限：1.cookie相对不是太安全，容易被盗用导致cookie欺骗 2.单个cookie的值最大只能存储4k 3.每次请求都要进行网络传输，占用带宽
6.session是将用户的会话数据存储在服务端，没有大小限制，通过一个session_id进行用户识别，PHP默认情况下session id是通过cookie来保存的，因此从某种程度上来说，seesion依赖于cookie。但这不是绝对的，session id也可以通过参数来实现，只要能将session id传递到服务端进行识别的机制都可以使用session。
7.session会自动的对要设置的值进行encode与decode，因此session可以支持任意数据类型，包括数据与对象等。
8.默认情况下，session是以文件形式存储在服务器上的，因此当一个页面开启了session之后，会独占这个session文件，这样会导致当前用户的其他并发访问无法执行而等待。可以采用缓存或者数据库的形式存储来解决这个问题

####有关正则表达式
1.PCRE库函数中，正则匹配模式使用分隔符与元字符组成，分隔符可以是非数字、非反斜线、非空格的任意字符。经常使用的分隔符是正斜线(/)、hash符号(#) 以及取反符号(~)，例如：
/foo bar/
#^[^0-9]$#
~php~
2.如果模式中包含分隔符，则分隔符需要使用反斜杠（\）进行转义。/http:\/\//
如果模式中包含较多的分割字符，建议更换其他的字符作为分隔符，也可以采用preg_quote进行转义。
$p = 'http://';
$p = '/'.preg_quote($p, '/').'/';
echo $p;
3.分隔符后面可以使用模式修饰符，模式修饰符包括：i, m, s, x等，例如使用i修饰符可以忽略大小写匹配：
$str = "Http://www.imooc.com/";
if (preg_match('/http/i', $str)) {
  echo '匹配成功';
}
4.正则表达式中具有特殊含义的字符称之为元字符，常用的元字符有：
\ 一般用于转义字符
^ 断言目标的开始位置(或在多行模式下是行首)
$ 断言目标的结束位置(或在多行模式下是行尾)
. 匹配除换行符外的任何字符(默认)
[开始字符类定义
] 结束字符类定义
| 开始一个可选分支
(子组的开始标记
) 子组的结束标记
? 作为量词，表示 0 次或 1 次匹配。位于量词后面用于改变量词的贪婪特性。 (查阅量词)
* 量词，0 次或多次匹配
+ 量词，1 次或多次匹配
{ 自定义量词开始标记
} 自定义量词结束标记
5.元字符具有两种使用场景，一种是可以在任何地方都能使用，另一种是只能在方括号内使用，在方括号内使用的有：
\ 转义字符
^ 仅在作为第一个字符(方括号内)时，表明字符类取反
- 标记字符范围
其中^在反括号外面，表示断言目标的开始位置，但在方括号内部则代表字符类取反，方括号内的减号-可以标记字符范围，例如0-9表示0到9之间的所有数字。
6.方括号内的\s匹配任意的空白符，包括空格，制表符，换行符。[^\s]+表示一次或多次匹配非空白符。
方括号内的\w匹配字母或数字或下划线
方括号内的\d匹配数字
7.正则表达式中每个元字符匹配一个字符，当使用+之后将会变的贪婪模式，它将匹配尽可能多的字符，但使用问号?字符时，它将尽可能少的匹配字符，既是懒惰模式。
8.括号的作用?

index.php
```php
<?php
//请修改变量p的正则表达式，使他能够匹配str中的姓名
$p = '/(\w+\s\w+)/';
// $p = '/\w+\s\w+/';
$str = "name:steven jobs";
preg_match($p, $str, $match);
var_dump($match);

echo "match[1]:".$match[1]; //结果为：steven jobs
```
运行成功
```
array(2) {
  [0]=>
  string(11) "steven jobs"
  [1]=>
  string(11) "steven jobs"
}
match[1]:steven jobs
```

index.php
```php
<?php
//请修改变量p的正则表达式，使他能够匹配str中的姓名
// $p = '/(\w+\s\w+)/';
$p = '/\w+\s\w+/';
$str = "name:steven jobs";
preg_match($p, $str, $match);
var_dump($match);

echo "match[1]:".$match[1]; //结果为：steven jobs
```
运行成功
```
array(1) {
  [0]=>
  string(11) "steven jobs"
}
match[1]:
```

####对象的高级特性
对象比较，当同一个类的两个实例的所有属性都相等时，可以使用比较运算符==进行判断，当需要判断两个变量是否为同一个对象的引用时，可以使用全等运算符===进行判断。
```php
class Car {
}
$a = new Car();
$b = new Car();
if ($a == $b) echo '==';   //true
if ($a === $b) echo '==='; //false
```
对象复制，在一些特殊情况下，可以通过关键字clone来复制一个对象，这时__clone方法会被调用，通过这个魔术方法来设置属性的值。
```php
class Car {
    public $name = 'car';
    public function __clone() {
        $obj = new Car();
        $obj->name = $this->name;
    }
}
$a = new Car();
$a->name = 'new car';
$b = clone $a;
var_dump($b);
```
对象序列化，可以通过serialize方法将对象序列化为字符串，用于存储或者传递数据，然后在需要的时候通过unserialize将字符串反序列化成对象进行使用。
```php
class Car {
    public $name = 'car';
}
$a = new Car();
$str = serialize($a); //对象序列化成字符串
echo $str.'<br>';
$b = unserialize($str); //反序列化为对象
var_dump($b);
```

####重载
PHP中的重载指的是动态的创建属性与方法，是通过魔术方法来实现的。属性的重载通过__set，__get，__isset，__unset来分别实现对不存在属性的赋值、读取、判断属性是否设置、销毁属性。
```php
class Car {
    private $ary = array();
    public function __set($key, $val) {
        $this->ary[$key] = $val;
    }
    public function __get($key) {
        if (isset($this->ary[$key])) {
            return $this->ary[$key];
        }
        return null;
    }
    public function __isset($key) {
        if (isset($this->ary[$key])) {
            return true;
        }
        return false;
    }
    public function __unset($key) {
        unset($this->ary[$key]);
    }
}
$car = new Car();
$car->name = '汽车'; //name属性动态创建并赋值
echo $car->name;
```
方法的重载通过__call来实现，当调用不存在的方法的时候，将会转为参数调用__call方法，当调用不存在的静态方法时会使用__callStatic重载。
```php
class Car {
    public $speed = 0;
    public function __call($name, $args) {
        if ($name == 'speedUp') {
            $this->speed += 10;
        }
    }
}
$car = new Car();
$car->speedUp(); //调用不存在的方法会使用重载
echo $car->speed;
```

####有关访问控制
访问控制通过关键字public，protected和private来实现。
被定义为公有的类成员可以在任何地方被访问
被定义为受保护的类成员则可以被其自身以及其子类和父类访问
被定义为私有的类成员则只能被其定义所在的类访问

####有关this和self
self::$speed;静态属性用这个
$this->name;普通属性用这个

this就是指向当前对象实例的指针，不指向任何其他对象或类。
self是指向类本身，也就是self是不指向任何已经实例化的对象，一般self使用来指向类中的静态变量。假如我们使用类里面静态（一般用关键字static）的成员，我们也必须使用self来调用。还要注意使用self来调用静态变量必须使用:: (域运算符号)

####构造函数私有化
如果构造函数定义成了私有方法，则不允许直接实例化对象了，这时候一般通过静态方法进行实例化，在设计模式中会经常使用这样的方法来控制对象的创建，比如单例模式只允许有一个全局唯一的对象。
```php
class Car {
    private function __construct() {
        echo 'object create';
    }

    private static $_object = null;
    public static function getInstance() {
        if (empty(self::$_object)) {
            self::$_object = new Car(); //内部方法可以调用私有方法，因此这里可以创建对象
        }
        return self::$_object;
    }
}
//$car = new Car(); //这里不允许直接实例化对象
$car = Car::getInstance(); //通过静态方法来获得一个实例
```

####foreach
index.php
```php
<?php
$students = array(
'2010'=>'令狐冲',
'2011'=>'林平之',
'2012'=>'曲洋',
);//10个学生的学号和姓名，用数组存储

//使用循环结构遍历数组,获取学号和姓名
foreach($students as  $i => $v)
{
    echo $i."->".$v;//index->value
    echo "<br />";
}
foreach($students as  $v)
{
    echo $v;//输出（打印）姓名
    echo "<br />";
}
?>
```
运行成功
```
2010->令狐冲
2011->林平之
2012->曲洋
令狐冲
林平之
曲洋
```

####获取时间例子
index.php
```php
<?php
date_default_timezone_set('Asia/ShangHai');
$today = date('m-d',time());//获取当天日期
$birthday = "03-12";//生日
$money = 200;//消费金额
$discount = 0.8;//八折优惠
if($today == $birthday){
    $money = $money * $discount;
}else{
$money = $money * 1;
}
echo $today;
echo "<br />";
echo $money;
?>
```
运行成功
```
03-12
160
```

####一个算术例子
index.php
```php
<?php
$maxLine = 4; //每排人数
$no = 16;//学生编号
$line = ceil($no/$maxLine);
$row = $no%$maxLine?$no%$maxLine:$maxLine;
echo "编号".$no."的座位在第".$line."排第"
    .$row."个位置";
?>
```
运行成功
编号16的座位在第4排第4个位置

####PHP中的错误控制运算符
PHP中提供了一个错误控制运算符"@"，对于一些可能会在运行过程中出错的表达式时，我们不希望出错的时候给客户显示错误信息，这样对用户不友好。于是，可以将@放置在一个PHP表达式之前，该表达式可能产生的任何错误信息都被忽略掉；
如果激活了track_error（这个玩意在php.ini中设置）特性，表达式所产生的任何错误信息都被存放在变量$php_errormsg中，此变量在每次出错时都会被覆盖，所以如果想用它的话必须尽早检查。
需要注意的是：错误控制前缀"@"不会屏蔽解析错误的信息，不能把它放在函数或类的定义之前，也不能用于条件结构例如if和foreach等。

index.php
```php
<?php
ini_set('track_errors', 1);
$conn = @mysql_connect("localhost"
    ,"username","passw");
echo "出错了，错误原因是：\n".$php_errormsg
    ;
?>
```
运行成功
```
出错了，错误原因是：
mysql_connect(): No such file or direc
```

####1=="sdf"
index.php
```php
$a = 1;
$b = "1";
var_dump($a == $b);
echo "<br />";
var_dump($a === $b);
echo "<br />";
var_dump($a != $b);
echo "<br />";
var_dump($a <> $b);
echo "<br />";
var_dump($a !== $b);
echo "<br />";
var_dump($a < $b);
echo "<br />";

$c = 5;
var_dump($a < $c);
echo "<br />";
var_dump($a > $c);
echo "<br />";
var_dump($a <= $c);
echo "<br />";
var_dump($a >= $c);
echo "<br />";
var_dump($a >= $b);
echo "<br />";
```
运行成功
```
bool(true)
bool(false)
bool(false)
bool(false)
bool(true)
bool(false)
bool(true)
bool(false)
bool(true)
bool(false)
bool(true)
```

####定义常量
获取常量的值的时候没有$符号，可以直接使用常量名或者使用constant()函数
index.php
```php
<?php
$p = "PII";
define("PI",3.14);
define($p,3.14);
echo PI;
echo "<br />";
echo constant("PI");
echo "<br />";
echo PII;
?>
```
运行成功
```
3.14
3.14
3.14
```

####PHP第二种特殊类型—空类型
NULL（NULL）：NULL是空类型，对大小写不敏感，NULL类型只有一个取值，表示一个变量没有值，当被赋值为NULL，或者尚未被赋值，或者被unset()，这三种情况下变量被认为为NULL。

####PHP第一种特殊类型—资源
资源（resource）：资源是由专门的函数来建立和使用的，例如打开文件、数据连接、图形画布。我们可以对资源进行操作（创建、使用和释放）。任何资源，在不需要的时候应该被及时释放。如果我们忘记了释放资源，系统自动启用垃圾回收机制，在页面执行完毕后回收资源，以避免内存被消耗殆尽。

####
当双引号中包含变量时，变量会与双引号中的内容连接在一起；
当单引号中包含变量时，变量会被当做字符串输出。
index.php
```php
<?php
$love = "I love you!";
$string1 = "$love";
$string2 = '$love';
echo $string1;
echo "<br />";
echo $string2;
?>
```
运行成功
```
I love you!
$love
```

####连接符
字符串用单引号、双引号括起来
<?php echo 'Hi,'.'imooc!';?>
注意:连接符可以连接多个字符串，上面的例子就是把一个字符串拆分为二个小字符串，然后用一个连接符连起来。
在PHP中字符串连接符是用点（.）来表示的，这一点比较特殊，其它语言中是用加号（+）来表示的，比如：JavaScirpt、Asp、C。

####有关?>的省略
在页面中编写PHP代码写在<?php?>标签之间，但注意后面的?>是可以省略的。
注意：如果全是PHP代码的话是可以省略的，如果是加在其他的代码中的话不能省略，是说明加入了一段PHP代码,到?>结束;

####
php.net下载php安装包后的php是不能跟Nginx一起工作的，只能跟Apache一起工作。如果要用Nginx作为webserver的话需要另外安装FPM(FastCGI进程管理器)。

####服务器多选centos、readhat而不是Ubuntu
前两者有10年生命周期，后者5年，在这之后不再更新 容易出现安全性问题。

####ubuntu下安装apache+php+mysql
mysql启动失败
```
andbase@vm-10-58-57-52:~$ sudo service mysql start
start: Job failed to start
andbase@vm-10-58-57-52:~$ service mysql status
mysql stop/waiting
andbase@vm-10-58-57-52:~$ service mysql start
start: Rejected send message, 1 matched rules; type="method_call", sender=":1.73" (uid=1002 pid=21041 comm="start mysql ") interface="com.ubuntu.Upstart0_6.Job" member="Start" error name="(unset)" requested_reply="0" destination="com.ubuntu.Upstart" (uid=0 pid=1 comm="/sbin/init ")
andbase@vm-10-58-57-52:~$
```
sudo cat /var/log/mysql/error.log 查看log显示写/tmp/xxx失败，运行在公司虚拟机上，/tmp是个软链接，权限都是root 777看着应该没问题。
修改/etc/mysql/my.conf中tmpdir /tmp 为tmpdir /var/lib/mysql然后启动就成功了。
```
190314 10:17:49 [Warning] Using unique option prefix myisam-recover instead of myisam-recover-options is deprecated and will be removed in a future release. Please use the full name instead.
190314 10:17:49 [Note] Plugin 'FEDERATED' is disabled.
190314 10:17:49 InnoDB: The InnoDB memory heap is disabled
190314 10:17:49 InnoDB: Mutexes and rw_locks use GCC atomic builtins
190314 10:17:49 InnoDB: Compressed tables use zlib 1.2.8
190314 10:17:49 InnoDB: Using Linux native AIO
/usr/sbin/mysqld: Can't create/write to file '/tmp/ib8aAIF5' (Errcode: 13)
190314 10:17:49  InnoDB: Error: unable to create temporary file; errno: 13
190314 10:17:49 [ERROR] Plugin 'InnoDB' init function returned error.
190314 10:17:49 [ERROR] Plugin 'InnoDB' registration as a STORAGE ENGINE failed.
190314 10:17:49 [ERROR] Unknown/unsupported storage engine: InnoDB
190314 10:17:49 [ERROR] Aborting

190314 10:17:49 [Note] /usr/sbin/mysqld: Shutdown complete
```